## **GETPASS**

Vulnerable to internal buffer overflows

Sean Barnum, Cigital, Inc. [vita<sup>1</sup>]

Copyright © 2007 Cigital, Inc.

2007-03-23

## Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 3863 bytes

Attack Category	Malicious In	Malicious Input			
Vulnerability Category	Buffer Overflow				
	• Input source (not really attack)				
	Unconditional				
Software Context	String Mana	String Management			
Location					
Description	Some versions of getpass() allow overflow of an internal buffer.				
	password from the terminated string appear on the scriproblem, but that In some implement a maximum lenger.	The getpass function is designed to accept a password from the console, which is a null-terminated string. The echo is off so it will not appear on the screen. It can lead to a buffer overflow problem, but that is very implementation dependent. In some implementations of the function, there is a maximum length defined for the password, and in other implementation, the password can be of arbitrary length.			
APIs		Function Name Comm		ents	
	getpass				
Method of Attack	-	Password entry is controlled by the user, who could potentially mount a buffer overflow attack.			
Exception Criteria					
Solutions	Solution Applicability	Solution Description		Solution Efficacy	
	Whenever password must be read from console.	Consult bug/ security notices relevant to the particular implementation of getpass(). Consider use of readpassphrase()		Effective if implementation is sound, or if one can substitute a function with a sound implementation.	

<sup>1.</sup> http://buildsecurityin.us-cert.gov/bsi/about\_us/authors/35-BSI.html (Barnum, Sean)

GETPASS 1

	on platf that sup		
Signature Details	char *getpass(const char* prompt);		
<b>Examples of Incorrect Code</b>	<pre>char *passwd = getpass("Enter password:");</pre>		
<b>Examples of Corrected Code</b>	<pre>char passwd[passwdBuffSize]; if (!readpassphrase("Enter password:", passwd, passwdBuffSize, 0)) { handleError(); }</pre>		
Source Reference	• Viega, John & McGraw, Gary. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, p. 148.		
Recommended Resource			
Discriminant Set	<b>Operating System</b>	• Windows	
	Languages	• C • C++	

## Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about "Fair Use," contact Cigital at copyright@cigital.com<sup>1</sup>.

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

GETPASS 2

<sup>1.</sup> mailto:copyright@cigital.com